Communication Between Dialogs

Event-driven applications require communication between various dialogs. New dialogs must be opened, and information must be exchanged between existing dialogs.

This section describes the various methods for communication between dialogs.

- The Standard Interface
- Calling a Dialog

The Standard Interface

All dialogs implemented using frames have a standard interface. Parameters are defined in the parameter data area ZXXREC0A. All variables of this parameter data area are collected within the group PZ_RECEIVE.

Standard Interface Structure

The standard interface is divided into two areas, PZ_STRUCTURE and PZ_DATA.

The group PZ_STRUCTURE contains all variables related to application control. These variables are used primarily by the frames.

The array PZ_DATA (A100/1: 40) is used for the transfer of technical data between dialogs. For example, an array #PZ_DATA (A100/1:n) can be defined and then redefined according to specific requirements. PZ_DATA is not used by the frames.

Example:

```
01 #PZ_DATA (A100/1:2)
01 REDEFINE #PZ_DATA
02 #COMMENT (A250)
02 #NAME    (A30)
.
.
   MOVE '   ' TO #COMMENT
   MOVE '   ' TO #NAME
   MOVE #PZ_DATA (1:2) to PZ_DATA (1:2)
```

Local Copy of the Interface

The parameter values are no longer available following the processing of the EVENT or OPEN DIALOG statements to which they are passed. Therefore, those parameter values which are required for a longer period of time must be saved as local variables. A local copy of the parameter data area (Natural object ZXXLOC0A) is available for this purpose. These variables are collected in group PZ_LOCAL.

During execution of internal commands, the frames transfer the relevant parameters from the group PZ_RECEIVE into the group PZ_LOCAL.

Communication Using User-Defined Events

User-defined events can be added in the dialog editor for communication between dialogs (see the *Natural User's Guide*). This is necessary if customized processes are to be executed.

Communication using Pre-Defined Event Z_CMD_EXEC

Using the pre-defined event Z_CMD_EXEC, it is possible for another dialog to process certain commands. The following suggested code can be used to do so (appropriate modifications must be made based on specific requirements):

```
MOVE ....     ..........TO PZ_LOCAL....
MOVE 'command'           TO PZ_LOCAL.PZ_CMD_ID
MOVE LZ_CMD_TYPE_INT     TO PZ_LOCAL.PZ_CMD_TYPE
SEND EVENT 'Z_CMD_EXEC' TO dialog-id
 WITH PZ_LOCAL
```

You can send any information via group PZ_LOCAL which contains PZ_DATA.

Calling a Dialog

A dialog call can be performed in various ways. The first dialog call occurs using an OPEN DIALOG statement. The communication between existing dialogs is accomplished using the SEND EVENT statement.

The frames perform for the most part the communication between dialogs. There are, however, situations in which communication must be implemented within customizable components. For example:

- Browsers and entry dialogs for functions can be started using the corresponding commands. Subdialogs can also be opened using commands.
- The opening of modal windows and key dialogs for selection of foreign keys must be individually coded.
- The exchange of information between dialogs.

Communication with Subdialogs

The subroutine Z_ASSIGN_SUBDIALOG in the maintain dialog is used to specify any subdialogs the maintain dialog calls. You must modify the suggested code to specify the module name of each subdialog and the local command ID you are associating with it and the total number of subdialogs associated with the maintain dialog. You must also associate the local command with a push button and define the command ID you specify in the application shell.

The actual communication between a subdialog and a maintain dialog is carried out by the internal frame logic. When the local command is invoked, either the corresponding dialog is opened, or, if it is already open, it will receive the focus.

After the subdialog has been opened, there can be further communication between the subdialog and the maintain dialog.

The frame logic for the subdialog informs the maintain dialog whenever data contained within the subdialog has been modified.

The frame logic of the subdialog or maintain dialog advises of the execution of commands which have an impact on the other dialog. The frame descriptions indicate which commands cause this type of communication.

Controlling a Subdialog from another Subdialog

If a subdialog is to be controlled from another subdialog, the command assigned in the maintain dialog must be passed from the subdialog to the maintain dialog. For this purpose, the following suggested code must be added and adapted to the subroutine Z_CUSTOM_CMD in the subdialog:


```
IF LZ_STD.LZ_FRAME_CMD_ID EQ 'command'
  MOVE LZ_STD.LZ_FRAME_CMD_ID    TO  PZ_LOCAL.PZ_CMD_ID
  MOVE LZ_STD.LZ_FRAME_CMD_TYPE  TO  PZ_LOCAL.PZ_CMD_TYPE
  SEND EVENT 'Z_CMD_EXEC'        TO  PZ_LOCAL.PZ_MAIN_DLG
   WITH PZ_LOCAL
END_IF
```

Foreign Key Selection/Active Help

The key dialog for input/output of primary keys is opened by the frame of the maintain dialog. This is done during the processing of standard commands whenever the input of a key value is required.

If a key dialog is opened to select a foreign key, the following suggested code can be added and adapted:

```
MOVE LZ_KEY_TYPE_FOREIGN  TO  LZ_STD.LZ_KEY_TYPE
MOVE #field               TO  PZ_LOCAL.PZ_SEL_KEY

OPEN DIALOG 'xxxKD0&D'  #DLG$WINDOW WITH PZ_LOCAL
```

When a key value is selected, the component Z_RECEIVE_KEY is executed in the calling dialog and the key value can be transferred to the corresponding field.

Note:
If from one dialog, selection help for various fields is to be called, the field for which selection help is to be opened must be marked with a user-defined indicator. During the transfer of the key value in component Z_RECEIVE_KEY, the corresponding field must be passed based on the identifier.

Calling Modal Windows

Modal windows are not opened by frames. The open dialog can be coded either in an event handler or in a specialized component. The following suggested code can be added and adapted:

```
MOVE data               TO  PZ_LOCAL.PZ_DATA(1:n)

OPEN DIALOG 'xxxMW0&D#  #DLG$WINDOW WITH PZ_LOCAL
```

Following the modification confirmation in the modal window, the component Z_RECEIVE_DATA is executed in the output dialog. The modified data from the array PZ_RECEIVE.PZ_DATA can then be transferred to the corresponding local variables.

Note:
If from one dialog, multiple modal windows are to be called, the window to be opened must be marked with a user-defined indicator. This indicator is also relevant during data transfer.

Commands for Opening a Dialog

The opening of certain dialogs can be invoked using a special type of command as defined in the *Natural Application ShellManual*. The frames of the dialog in which the commands are to be invoked open automatically the desired dialog. They need not be programmed.

Starting an Application

A dialog for icon-based navigation can be started with the command type "Start an application".

Starting a Browser

Browse functions can be started using the command type "Start a browser".

Starting a Function

Functions can be started using the command type "Start a function". A key is provided using the variable PZ_LOCAL.PZ_SEL_KEY. This is used for the function start if the switch PZ_LOCAL.PZ_KEY_FILLED is set to TRUE.

In maintain dialogs, a new function with the same Action and Object Type is started using the command Z_OPEN.

In maintain dialogs, for the action type New, a new function with the same object type and the selected action is started.

For browse dialogs, for the commands of type Action, a function for each selected data record is started.

For a description of how to start applications, browse functions and other functions, see Starting a Dialog (Application, Function, Browse).

